UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
UNIDADE ACADÊMICA DE SISTEMAS E COMPUTAÇÃO
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

MAIANA BRITO

ANALISANDO A APRENDIZAGEM DE DIFERENTES PARADIGMAS DE PROGRAMAÇÃO NO CONTEXTO DE UMA DISCIPLINA

CAMPINA GRANDE - PB

2019

MAIANA BRITO

ANALISANDO A APRENDIZAGEM DE DIFERENTES PARADIGMAS DE PROGRAMAÇÃO NO CONTEXTO DE UMA DISCIPLINA

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharela em Ciência da Computação.

Orientador: Professor Dr. Everton Leandro Galdino Alves.

CAMPINA GRANDE - PB

2019

B862a Brito, Maiana.
Analisando a aprendizagem de diferentes paradigmas de programação no contexto de uma disciplina. / Maiana Brito. – 2019.

11 f.

Orientador: Prof. Dr. Everton Leandro Galdino Alves.
Trabalho de Conclusão de Curso - Artigo (Curso de Bacharelado em Ciência da Computação) - Universidade Federal de Campina Grande; Centro de Engenharia Elétrica e Informática.

1. Aprendizagem - programação. 2. Linguagens de programação. 3. Paradigma de programação. 4. Estudo de programação. I. Alves, Everton Leandro Galdino. II. Título.

CDU:004(045)

Elaboração da Ficha Catalográfica:

Johnny Rodrigues Barbosa
Bibliotecário-Documentalista
CRB-15/626

MAIANA BRITO

ANALISANDO A APRENDIZAGEM DE DIFERENTES PARADIGMAS DE PROGRAMAÇÃO NO CONTEXTO DE UMA DISCIPLINA

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharela em Ciência da Computação.

BANCA EXAMINADORA:

Professor Dr. Everton Leandro Galdino Alves
Orientador – UASC/CEEI/UFCG

Professor Dr. Hyggo Oliveira de Almeida
Examinador – UASC/CEEI/UFCG

Professor Dr. Tiago Lima Massoni
Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 25 de novembro 2019.

CAMPINA GRANDE - PB

Analisando a Aprendizagem de Diferentes Paradigmas de Programação no Contexto de uma Disciplina

Trabalho de Conclusão de Curso

Maiana Brito
maiana.brito@ccc.ufcg.edu.br
Universidade Federal de Campina Grande
Campina Grande, Paraíba

Everton L. G. Alves*
everton@computacao.ufcg.edu.br
Universidade Federal de Campina Grande
Campina Grande, Paraíba

RESUMO

Um paradigma reflete um conjunto de características referentes a um estilo de programação. Linguagens de programação são classificadas de acordo com os paradigmas que aplicam. Dessa forma, o ensino de paradigmas é muito importante na formação de profissionais que trabalham na área de programação. Diante disso, o curso de Ciência da Computação na UFCG aborda o ensino de diversos paradigmas na disciplina Paradigmas de Linguagem de Programação (PLP) que combine aulas teóricas e exercícios práticos. O presente trabalho busca analisar o histórico de resposta dos alunos de PLP para exercícios considerando três diferentes paradigmas (imperativo, funcional e lógico), a fim de identificar padrões e identificar as dificuldades. Para este estudo, utilizou-se dados históricos das atividades práticas de quatro períodos letivos (2017.2, 2018.1, 2018.2 e 2019.1), avaliando parâmetros como: quantidade de acertos e tentativas de uma atividade, e o tempo que o aluno precisa até obter o sucesso. Analisando os dados descobrimos que nas questões dos paradigmas lógico e funcional, são necessárias poucas tentativas para obter o sucesso, já o imperativo observou-se que normalmente mais tentativas são necessárias. Entretanto, o tempo se mostrou equivalente para os paradigmas. Diante disso, não foi possível definir um paradigma que se mostrasse problemático, em todas as situações, sendo necessário mais dados e/ou outras abordagens.

1 INTRODUÇÃO

Nos dias atuais, independente da área de atuação, é crescente a necessidade do tratar com tecnologia, e com isso o interesse na área de programação vêm crescendo. Nesse sentido, vários estudos têm surgido para identificar os desafios envolvidos no processo de ensino/aprendizagem de programação. Em [6], o autor afirma que "Os desafios mais comuns se encontram nas dificuldades de abstração e resolução de problemas", interligado à isso, pode-se pontuar a dificuldade em desenvolver o raciocínio lógico por trás do funcionamento da programação. O que leva ao questionamento de qual a melhor linguagem e estilo de programação para aprender a programar?

A discussão que permeia a escolha da linguagem de programação se justifica pois cada linguagem possui seu conjunto de características próprias. Algumas, possuem um nível maior de abstração, se assemelhando à linguagem natural, exemplo Python. Outras, estabelecem uma proximidade maior com a linguagem de máquina, por exemplo, Assembly.

Além da sintaxe, cada linguagem aborda um conjunto de características referentes ao estilo de programação. Esses estilos são chamados de Paradigmas de Linguagem de Programação. Por exemplo, o paradigma funcional tem como sua essência o uso de funções, que se baseiam em conceitos de funções matemáticas, sem efeitos colaterais.

Existem diversos paradigmas como: Orientado à Objeto, Lógico, Declarativo, entre outros. Cada paradigma detém suas próprias especificidades e características. Em [15], diversos paradigmas são discutidos, bem como as linguagens que utilizam seus conceitos. Por exemplo, o paradigma orientado a eventos se baseia na ocorrência de eventos externos e tem como exemplos de linguagens que o implementam Delphi, Python e Java. Ainda em Tedesco et al. [15], ele aponta que certos paradigmas são mais adequados para resolução de problemas de contextos específicos. Ainda existem aquelas linguagens que combinam características de diferentes paradigmas (linguagens multiparadigmas). Desse modo, ao programador, é imprescindível conhecer bem os diversos paradigmas, seus fundamentos, bem como o contexto do problema que irá tratar.

Diante disso, o curso de Ciência da Computação da UFCG reconhece a importância desse tópico. É sabido que grande parte dos alunos do curso escolhem se especializar em desenvolvimento de software. Desta forma, é imprescindível estes tenham contato com diferentes paradigmas de programação. Para esta finalidade, foi criada a disciplina Paradigmas de Linguagem da Programação (PLP), na qual, além de aspectos básicos de projeto de linguagens de programação, são abordados três diferentes paradigmas: Imperativo, Funcional e Lógico. Estes são reconhecidamente alguns dos paradigmas mais conhecidos e utilizados na prática [1]. A disciplina é conduzida seguindo uma metodologia que combina aulas teóricas e práticas. Nas práticas, os alunos precisam resolver problemas de programação utilizando, em momentos diferentes, linguagens de paradigmas diferentes. Desta forma, o objetivo da disciplina não é só que os alunos entendam os conceitos dos paradigmas, mas que também aprendam a desenvolver programas corretos e segundo o estilo de cada paradigma. É importante ressaltar, que os três paradigmas em questão (imperativo, funcional e lógico), são introduzidos usando as linguagens de programação C, Haskell e Prolog, respectivamente.

Aprender diferentes paradigmas não é algo trivial. Porém, não sabemos se existe alguma relação padrão de dificuldade entre alunos e um determinado paradigma ou laboratório (grupo de exercícios). Diante disso, busca-se entender através de um estudo, se algum

paradigma representou maior dificuldade para os alunos, e se sim, o porquê? Para entender melhor esse cenário, foram utilizados dados históricos coletados da submissão dos alunos na disciplina ao longo de quatro períodos. Nosso objetivo, é tentar identificar padrões de comportamento que indiquem graus de dificuldade por parte dos alunos. Para avaliar dificuldade, foram definidos alguns critérios: se para algum paradigma os alunos necessitam de mais tentativas para encontrar a resposta correta, e o tempo entre a primeira tentativa até o acerto. Através desse estudo, pretende-se entender qual paradigma é o mais complexo para os alunos, e assim sugerir possíveis pontos de melhorias.

Com este trabalho, foi possível obter e analisar as respostas para as perguntas de pesquisa, que buscam caracterizar o que seria a dificuldade, entretanto não houve convergência nas três respostas para que se aponte o paradigma que apresente maior dificuldade.

2 FUNDAMENTAÇÃO TEÓRICA

O ensino de programação é um tema bastante discutido e é a base do nosso projeto, sobre este tema, diversas pesquisas foram desenvolvidas abordando tópicos específicos como: dificuldades para o ingresso e permanência de estudantes do sexo feminino na Ciência e Engenharia da Computação, abordada em [12]; a metodologia aplicada e interferência da mesma nos alunos, referindo por exemplo, ao índice de reprovação. Campos et al., 2010 [11] observou que com o uso da metodologia ERM2C obteve uma redução no índice de reprovação dos novatos de 37% para 18%.

Cada vez mais, o uso e a criação de novas ferramentas se fazem presente no processo de ensino, a exemplo de Heming et al., 2019 [4] que desenvolveu uma ferramenta a qual auxilia no entendimento e compreensão dos exercícios das disciplinas de algoritmos e programação. Além disso, a ferramenta fornece um ambiente de estudo fácil, rápido e simples.

É importante também buscar entender quais desafios são encontrados durante aprendizagem de programação para que assim alternativas sejam desenvolvidas. Em [9], o autor após sua análise, atribui como maiores dificuldades para os alunos o entendimento da sintaxe da linguagem e o desenvolvimento da lógica.

Trazendo para o contexto de paradigmas de programação, [10] explora o impacto dos paradigmas no ensino intermediário de programação. Em sua pesquisa ele adotou dois paradigmas: orientado a objeto e estruturado, através das linguagens de programação, Java e C, e apontou que o paradigma adotado pela linguagem pode simplificar a resolução dos exercícios propostos em Estrutura de Dados (a disciplina na qual ocorreu o estudo).

Como citado anteriormente, a metodologia de ensino interfere nos índices de aprovação de uma turma, o que está diretamente relacionado à aprendizagem. Uma metodologia comum em disciplinas de programação é utilizar atividades práticas desenvolvidas ao longo do período.

Berssanet et al., 2016 [8] observou que a prática favoreceu o aprendizado, isso devido ao fato de promover o feedback imediato que ajuda os alunos a desenvolver o raciocínio lógico e promove um melhor planejamento para implementação e/ou correções dos códigos, podendo contribuir principalmente para a motivação do aluno e também para a diminuição das abstrações presentes no processo. Além disso o autor conseguiu observar pontos que ao transmitir o conceito não foi dada a devida atenção.

Valentim analisou o ensino-aprendizagem do paradigma lógico, o qual diz que "tornar o conhecimento concreto por meio do papel", ou seja, desenvolver atividades auxilia no processo de pensar logicamente [7]

As atividades práticas desenvolvidas muitas vezes são assessoradas por ferramentas de correções automáticas. Essas ferramentas retornam aos alunos feedbacks instantâneos sobre suas soluções. De acordo com Gaudêncio et al., 2013 [5] os professores notaram grandes benefícios, como por exemplo, com os relatórios os mesmos possuem uma visualização dos alunos individualmente, possibilitando auxiliar em assuntos específicos como também o andamento da turma de forma geral.

Diante do contexto, nosso trabalho busca analisar dados históricos de uma disciplina que introduz diferentes paradigmas através do uso de ferramentas de feedback automático. Nesta análise, busca-se entender se os alunos possuem alguma dificuldade em paradigmas de programação específicos.

3 CONTEXTUALIZAÇÃO E METODOLOGIA

Nos primeiros períodos do curso de Ciência da Computação da Universidade Federal de Campina Grande (UFCG), os alunos possuem contato inicial com as disciplinas de desenvolvimento de software, trabalham linguagens específicas (atualmente Python e Java) e utilizam paradigmas de Linguagem de Programação de forma intuitiva. Somente na disciplina Paradigmas de Linguagens de Programação (PLP) os alunos são apresentados com detalhes aos diferentes paradigmas, entendendo de fato suas especificidades.

Na disciplina PLP, os alunos são apresentados à três paradigmas diferentes: imperativo, funcional e lógico. Para cada paradigma, os alunos são introduzidos aos principais conceitos e aplicações de cada paradigma, bem como são apresentados a linguagens que os implementam. Tendo em vista que os alunos já tiveram anteriormente contato com o paradigma imperativo (na disciplina Programação 1), os demais paradigmas têm um maior espaço no cronograma de aulas.

Buscando intensificar a aprendizagem, a metodologia da disciplina inclui a realização de diversas atividades práticas, os laboratórios. Ao longo do período, são aplicados cinco laboratórios com a seguinte divisão: **um** laboratório sobre o paradigma **imperativo**, utilizando a linguagem **C**, **dois** laboratórios sobre o paradigma **funcional**, utilizando a linguagem **Haskell**, e **dois** laboratórios sobre o paradigma **lógico**, utilizando a linguagem **Prolog**.

Nos laboratórios, conceitos básicos de programação são exercitados segundo cada paradigma. A Tabela 1 referencia quais conceitos são exercitados em cada laboratório:

Os laboratórios consistem de problemas práticos de programação, nos quais os alunos, em duas horas, devem desenvolver soluções para três questões. As atividades são disponibilizadas e avaliadas através da ferramenta Dirlididi[1]. Dirlididi é uma ferramenta e site para gerência de questões de programação que foi adaptada para ser usada para diferentes paradigmas. Nela, o professor hospeda questões e testes automáticos, os mesmos estão associados à cursos, assim como os alunos. A partir do código de cada questão (fornecido

[1] http://dirlididi.com/

C	• Entrada e Saída • Loops • Estrutura de Dados
Haskell Básico	• Entrada e Saída • Funções • Recursividade
Haskell Avançado	• Funções • Listas • Recursividade
Prolog Básico	• Entrada e Saída • Relações • Recursividade
Prolog Avançado	• Relações • Listas • Recursividade

Tabela 1: Assuntos aplicados/exercitados em cada laboratório

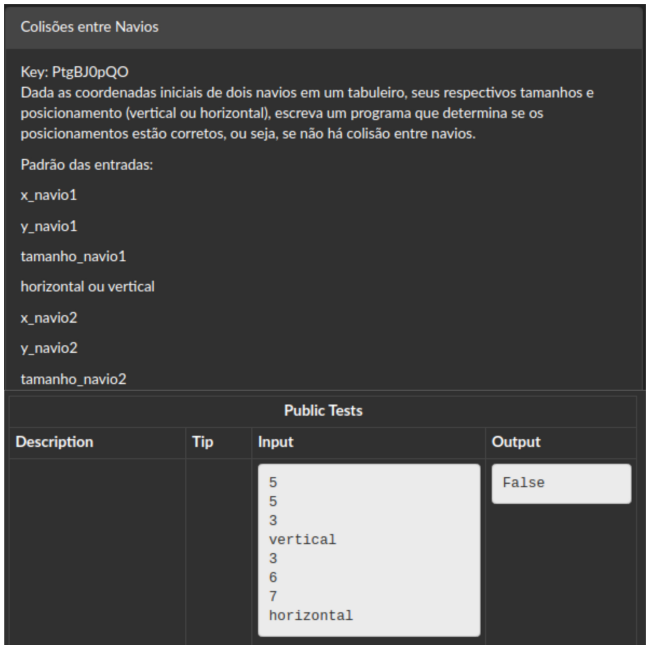

Figura 1: Imagem de uma questão

pelo professor), o aluno consegue acessar a atividade e submeter sua resposta. A Figura 1 exemplifica uma questão de laboratório no Dirlididi. Vale salientar que, apesar de cada laboratório conter um conjunto diferente de questões, o professor buscar equilibrar seus níveis de dificuldade, uma vez que o objetivo maior é exercitar o uso dos paradigmas.

Cada atividade dispõe de um anunciado, exemplos de entrada e saída esperados (os chamados **public tests**), e um conjunto de testes não visíveis para os alunos, porém seus resultados são visíveis. Uma vez que gerou seu código-resposta para uma dada questão do laboratório usando a linguagem exigida (C, Haskell, Prolog), o aluno pode executar seu código-resposta no servidor e obter o *feedback* dos testes adicionados pelo professor.

Na Figura 1. podemos ver um exemplo concreto de uma atividade, esta chama-se "Colisões entre Navios", o objetivo da mesma é a partir das dimensões e localização de dois navios, determinar se haverá colisão entre eles. Esta atividade foi aplicada no laboratório de Haskell Avançado no período 2017.2, ela contempla os assuntos de *funções*, *listas* e *recursividade*. Em uma possível solução para esta questão, o aluno precisará criar um conjunto de funções para validação horizontal da posição e validação vertical da posição. As posições podem ser tratadas através de uma lista, representando espaços vazios com um caractere e espaços ocupados com outro, e o preenchimento do mapa realizado utilizando uma função recursiva.

A ferramenta Dirlididi é usada no contexto de PLP para gerenciar os laboratórios e avaliar as respostas dos alunos. Vale salientar que a ferramenta mantém histórico de todas as submissões dos alunos, incluindo aquelas que não obtiveram sucesso.

Sabe-se que os diferentes paradigmas possuem estrutura e fundamentação divergentes entre si. Assim, o objetivo deste trabalho *é entender os padrões de comportamento dos estudantes quanto resolvendo problemas de programação em diversos paradigmas*. Para guiar a investigação, as seguintes perguntas de pesquisa foram elaboradas:

- RQ1: Qual(is) paradigma(s) requerem uma maior quantidade de submissões?
- RQ2: Para qual(is) paradigma(s) um estudante necessita de mais tentativas até obter o sucesso?
- RQ3: Para qual(is) paradigma(s) os alunos demoram mais tempo até obter o sucesso?

As três questões acima listadas proporcionam uma análise referente a dificuldade que os alunos têm em prover soluções corretas a exercícios de programação relacionados aos paradigmas. Uma maior quantidade de submissões de respostas normalmente está relacionada a maior dificuldade com o paradigma, uma vez que os alunos podem submeter várias tentativas de código até que todos os testes passem. Nesse sentindo RQ1 e RQ2 buscam analisar esse tópico, uma vez que avaliam a quantidade de submissões por paradigma e quantas submissões cada estudante normalmente precisa realizar até passar em todos os testes, respectivamente. Já RQ3, busca complementar essa investigação analisando o tempo médio que os alunos levam para chegar na solução correta para o problema.

Para responder essas perguntas, utilizamos os dados armazenados no Dirlididi da disciplina PLP de quatro períodos (2017.2 até 2019.1). Os dados se referem a todas as submissões realizadas pelos alunos, para cada submissão dispomos das seguintes informações:

- Código da atividade
- ID do aluno
- Resultado dos testes
- Horário da Submissão

Além das informações do Dirlididi, o professor da disciplina forneceu um banco de dados, com o código de todas as questões, os períodos em que as mesmas foram aplicadas, como também a indicação de quais laboratórios são relacionados a qual paradigma.

De posse dos dados, o próximo passo foi o tratamento dos mesmos, removendo todas submissões que não correspondem ao horário de aplicação dos laboratórios, como também submissões referentes aos professores e monitores da disciplina. Em seguida, classificou-se as submissões como corretas (aprovadas em todos os testes) ou não. Por fim, cada submissão foi associada à um paradigma, período e laboratório, através do código da atividade. A partir dos dados filtrados, foi realizado uma análise quantitativa de acordo com as seguinte métricas por atividade:

- M1: Quantidade total de submissões;
- M2: Coeficiente de submissões;
- M3: Duração (em segundos) que cada aluno investiu na resolução da atividade.

Com M1, é possível determinar a quantidade de submissões por atividade, e assim responder a RQ1. A métrica com a quantidade de submissões por atividade foi calculada através do somatório de todas as submissões associadas à uma atividade. Já M2, foi definida a fim de avaliar a quantidade **tentativas** para atingir sucesso em uma atividade, de modo geral. Esse coeficiente se refere à uma atividade e é calculado da seguinte forma:

$$coeficiente = (n_acertos_atividade)/(total_submissoes_atividade) \quad (1)$$

Interpretativamente, valores próximos a *um* para o coeficiente refletem que os alunos tiveram poucas tentativas a chegar na resposta correta, indicando baixa dificuldade. Já valores mais próximos a *zero*, indicam poucos acertos em meio a todas as submissões, refletindo uma maior dificuldade, uma vez que o número de tentativas mal sucedidas foi grande. Uma vez dito isso, com a M2 é possível responder a RQ2.

Para calcular a M3, foi criado um script[2] que verifica, para cada aluno, o tempo em segundos que o mesmo levou entre a primeira submissão **incorreta** de uma atividade X até a submissão **correta** da atividade X. Para os casos em que os alunos obtiveram sucesso já na primeira submissão, o tempo foi **zero**. Essa métrica foi criada com o objetivo de responder a RQ3. O script implementado gerou um dataframe com todas as atividades realizadas por cada aluno, e quanto tempo durou, na Figura 2, pode-se ver um exemplo dos dados coletados.

Todos dados e as análises geradas para desenvolver este trabalho estão presentes no nosso site[3].

4 RESULTADOS E DISCUSSÕES

A partir dos dados coletados, obtivemos informações das submissões de 488 alunos, em quatro períodos e 20 laboratórios aplicados. Podemos visualizar um sumário com as informações por período na Figura 3, e por paradigma na Figura 4.

[2]https://github.com/MaianaB/TCC_Analise_na_aprendizagem_de_diferentes_paradigmas/blob/master/utils/script_tempo.js
[3]https://github.com/MaianaB/TCC_Analise_na_aprendizagem_de_diferentes_paradigmas

cod_atividade	aluno	duracao	paradigma	laboratorio	periodo
MiQTU7m76	adson.silva@ccc.ufcg.edu.br	1382.503000	Lógico	PrologBasico	2017.2
QEDoVk8tE	adson.silva@ccc.ufcg.edu.br	875.283000	Lógico	PrologBasico	2017.2
RYIhBbemO2	adson.silva@ccc.ufcg.edu.br	0.000000	Lógico	PrologBasico	2018.1
PwDK8kgEa	aislan.brito@ccc.ufcg.edu.br	0.069005	Procedural	C	2018.1
L4o87izAG	aislan.brito@ccc.ufcg.edu.br	418.506000	Funcional	HaskellBasico	2018.1
S7I4AKn20	aislan.brito@ccc.ufcg.edu.br	0.000000	Funcional	HaskellBasico	2018.1
SVqGDYZsW	aislan.brito@ccc.ufcg.edu.br	226.285000	Funcional	HaskellBasico	2018.1
QaQSw1Z3Y	aislan.brito@ccc.ufcg.edu.br	0.000000	Funcional	HaskellAvancado	2018.1
S1y6iT1QO	aislan.brito@ccc.ufcg.edu.br	155.818000	Funcional	HaskellAvancado	2018.1

Figura 2: Amostra dos dados do tempo gasto em uma atividade por aluno

período <fctr>	n_alunos <int>	n_atividades <dbl>	n_submissoes <int>	n_acertos <int>	n_erros <int>
2017.2	187	13	1374	661	713
2018.1	144	14	1260	662	598
2018.2	93	15	1352	814	538
2019.1	64	15	1034	616	418

Figura 3: Sumário com informações agrupadas por período

paradigma <fctr>	n_atividades <int>	n_submissoes <int>	n_acertos <int>	n_erros <int>
Imperativo	12	1143	480	663
Funcional	24	1896	1179	717
Lógico	22	1981	1094	887

Figura 4: Sumário com informações agrupadas por paradigma

Para realizar a análise dos dados obtidos, utilizou-se a ferramenta RStudio [4], a qual, através da linguagem R, oferece pacotes para tratamento de dados, criação de métricas e visualização das informações.

Com o objetivo de estudar o comportamento das submissões, na Figura 5, podemos visualizar a quantidade de submissões por paradigma, e assim responder a **QR1**, com esta visualização pode-se dizer que o paradigma que possui mais submissões é o Lógico (1981 submissões).

Entretanto, esta é uma visualização muito simplista, pois é facilmente impactada por outliers, o que interfere nos valores finais. Desse modo, observar as atividades, sendo estas representadas pelos pontos, referentes à cada paradigma, proporciona uma melhor visualização (Figura 6).

Na Figura 6, foi utilizado a métrica M1 de cada atividade, em seguida as atividades foram agrupadas por paradigma, e exibidas em um boxplot, representação tradicional para distribuições, que fornece informações como mediana, valor mínimo, quartios, e a visualização de outliers.

Antes de qualquer resultado é necessário aplicar os testes estatísticos sobre os dados, e assim comprovar que os mesmo são diferentes entre si, para então construir as análises e responder as perguntas propostas. Inicialmente, foi aplicado o teste de Shapiro Wilk [3], este teste determina se os dados apresentam distribuição normal ou não. O teste define como hipótese nula (H0) a distribuição normal dos dados, e a partir do p-valor podemos rejeitar ou não a hipótese nula, caso o seu valor seja inferior à **0.05** rejeita-se H0, comprovando que os dados tem distribuição **não normal**.

[4]https://rstudio.com/

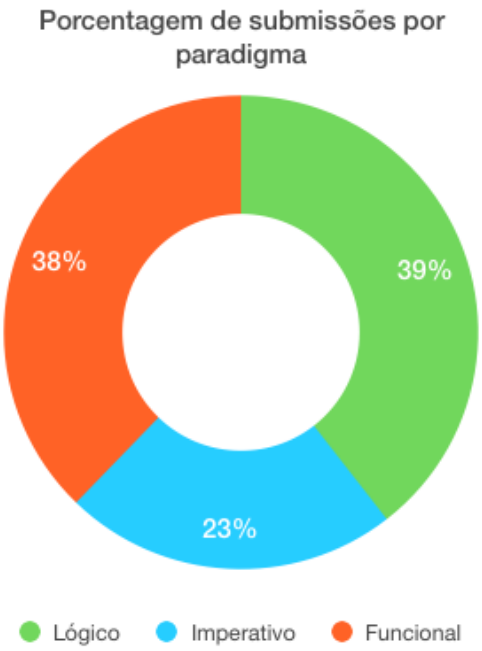

Figura 5: Gráfico dump com a quantidade de submissões por paradigma

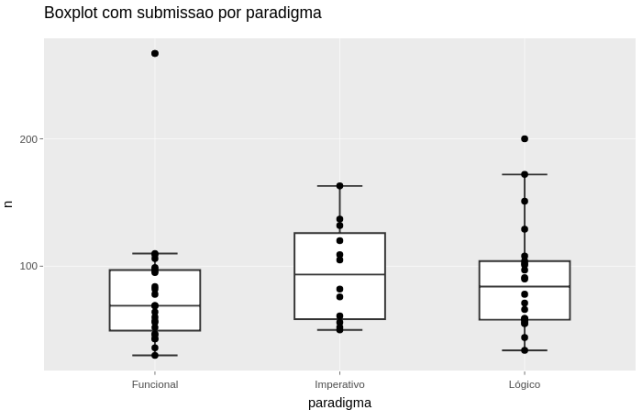

Figura 6: Boxplot da quantidade de submissões de atividades por paradigma

Ao efetuar o teste de Shapiro-Wilk sobre os dados das submissões, obteve-se o *p-valor igual à 9.9e-06*, que é um número inferior à 0.05, dessa forma temos que os dados **não** apresentam distribuição normal.

Para os casos em que os dados são não paramétricos, ou seja não possui distribuição normal, utiliza-se o teste de *Kruskal-Wallis*[13], o qual em sua hipótese nula (H0) define que as distribuições são estatisticamente equivalentes, já a hipótese alternativa é de que ao menos duas populações possuem funções de distribuições diferentes. Aplicou-se sobre os dados referentes aos *paradigmas* e *submissões*, o teste de Kruskall obteve como retorno o *p-value igual à 0.2517*, não rejeitando a hipótese nula. Diante disso, temos que **as distribuições são estatisticamente iguais**. O fato das distribuições serem iguais, significa que não é possível diferenciar entre elas e assim, classificar separadamente um paradigma. Desse modo, para **RQ1** *temos que, quanto ao número de submissões, todos os paradigmas se comportam da mesma forma estatisticamente.*

Embora, não seja possível apontar estatisticamente um paradigma exato para a **RQ1**, com o bloxpot consegue-se levantar informações interessantes, por exemplo, adotando a mediana como representante dos dados, temos que o paradigma Imperativo é aquele com maior quantidade de submissão (mediana de 93 submissões), entretanto, esse paradigma é abordado em apenas um laboratório, enquanto os outros são exercitados em dois laboratórios, ou seja, menos atividades. Com isso, percebemos que mesmo que os paradigmas Funcional e Lógico possuam mais atividades, o perfil das mesmas é de não possuírem tanto submissões, ao contrário das atividades do Paradigma Imperativo.

Outro ponto interessante é um outlier no paradigma Funcional, essa questão possui 267 submissões, a mesma foi exibida na sessão de metodologia (Figura 1). Quando buscamos mais detalhes sobre essa atividade, alguns alunos disseram não ter compreendido exatamente qual era a situação problema, e não conseguiram planejar corretamente a lógica para respondê-la, e na questão também era necessário fazer o uso de "listas de listas" (matrizes), o que se mostrou ser um grande desafio iniciantes em Haskell.

Seguindo com a investigação, a fim de responder RQ2 precisamos analisar os resultados de M2. Como visto anteriormente, a visualização dos dados considerando cada atividade agrupadas por paradigmas, é mais completa (Figura 8). Assim como aconteceu para os dados de submissões, foi necessário a realização de testes estatísticos a fim de estabelecermos conclusões a partir dos dados.

Para a métrica coeficiente (M2), o p-valor resultado do teste Shapiro-Walk foi de **0.8717**, dizendo que ele não rejeita a hipótese nula, e os dados apresentam **distribuição normal**. O próximo passo, foi a realização do teste que trata dados com distribuição normal, chamado Ttest. Assim como o Kruskall, o Ttest [2] também busca dizer se a distribuição dos dados são ou não diferentes. O p-valor obtido desse teste foi de i 1.052e-15, dessa forma rejeitamos a hipótese nula, que seria se eles possuem distribuição iguais.

Constatado que pelo menos duas distribuições são diferentes, é necessário realizar o teste par a par, para definir que todas apresentam distribuições diferentes, então aplicou-se o Ttest passando duas amostras, o mesmo valida a heterogeneidade de duas amostras, assumindo a hipótese nula que a distribuição de ambas as amostras é a mesma.

Aplicando o teste par a par por paradigmas encontramos os seguintes p-valores:

- 3.534e-06 para Imperativo e Funcional;
- 0.002975 para Imperativo e Lógico;
- 0.03049 Funcional e Lógico.

Em todos os casos o p-valor foi inferior à 0.05, rejeitando assim a hipótese nula, e confirmando que todos apresentam distribuições diferentes. Observando os p-valores, é possível montar um ranking, no qual o que possui o menor p-valor é aquele que se mostra

mais diferente do seu par, para este caso, o paradigma Imperativo estatisticamente é que se caracteriza mais diferente dos demais.

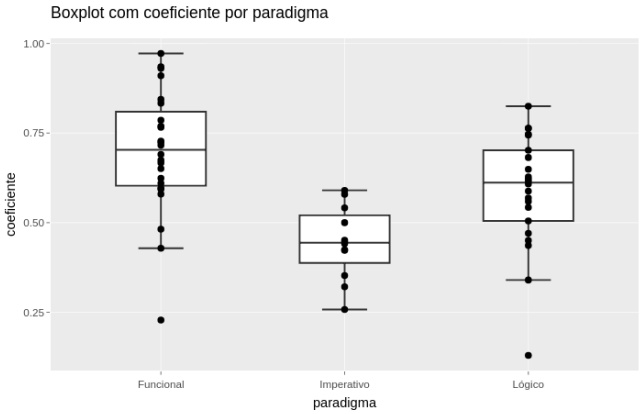

Figura 7: Boxplot do coeficiente de cada atividade agrupado por paradigma

Uma vez validado os dados, o próximo passo é analisar M2. Como dito anteriormente, a métrica M2 representa a tentativa, podemos percebemos que o paradigma funcional apresentou o comportamento de menos submissões necessárias para chegar no acerto, tendo sua distribuição de coeficiente indicando que a maior parte das submissões foram bem sucedidas. Já o paradigma imperativo apresentou grande quantidade de submissões, seu coeficiente teve como valor máximo (0.59), inferior as duas medianas dos outros paradigmas, e indica que foram necessárias várias tentativas até o acerto, respondendo a pergunta **RQ2**.

Ainda observando as métricas de **submissão** e **coeficiente**, conseguimos detalhar mais, observando os laboratórios especificamente (Figuras 8 e 9).

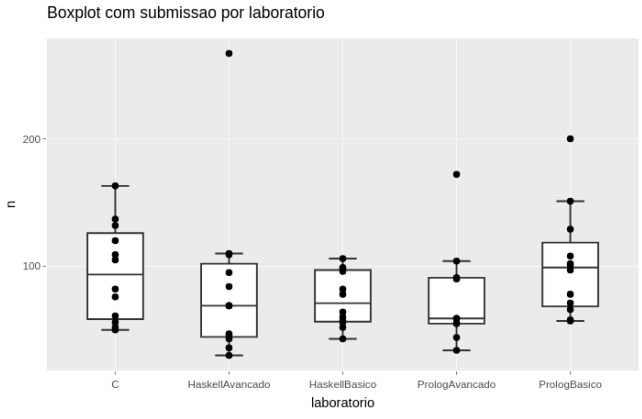

Figura 8: Boxplot com a quantidade de submissões e os laboratórios

Também foram aplicado os testes estatísticos, em ambos os casos submissões por laboratório e coeficiente por laboratório. No contexto de submissões, o teste de Kruskall manteve a H0 dizendo que

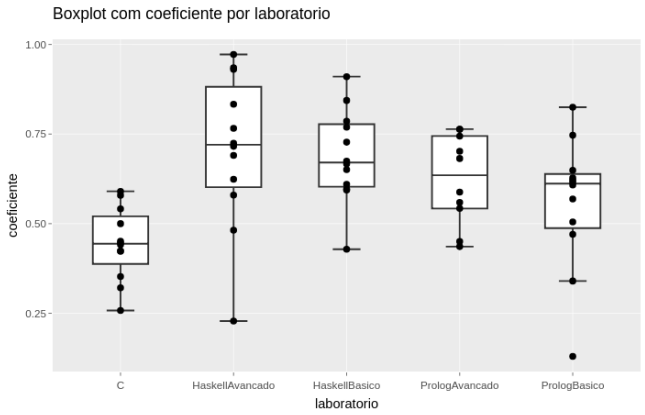

Figura 9: Boxplot com o coeficiente e os laboratórios

não há diferença na distribuição dos grupos. Já para o coeficiente, o Ttest provou que pelo menos duas distribuições são diferentes, aplicando Ttest par a par obtemos os seguintes p-valores:

- 4.271e-05 para C e Haskell Básico;
- 0.001378 para C e Haskell Avançado;
- 0.001924 para C e Prolog Avançado;
- 0.06039 para Haskell Básico e Prolog Básico;
- 0.07455 para C e Prolog Básico;
- 0.08177 para Haskell Avançado e Prolog Básico;
- 0.247 para Haskell Básico e Prolog Avançado;
- 0.2663 para Haskell Avançado e Prolog Avançado;
- 0.3427 Prolog Básico e Prolog Avançado;
- 0.7972 Haskell Básico e Haskell Avançado.

De acordo com o teste os agrupamentos só apresentam distribuições diferentes entre sí, quando seu p-valor é inferior à 0.05. Desse modo, apenas o laboratório de C que possui distribuição diferente dos demais. Observando o boxplot e levando sua mediana como representação, ele ainda possui o menor coeficiente. Diante disso, é possível responder **RQ2** indicando o paradigma **Imperativo** como aquele que é necessário uma quantidade maior submissões até o acerto.

Partindo para RQ3, nas figuras 10 e 11 visualizamos a distribuição do tempo investido em cada atividade por aluno, sendo cada ponto o tempo que um aluno X demorou para responder uma questão Y. Sobre esses dados, também realizamos os testes estatísticos, com isso foi definido que sua distribuição não é normal (Shapiro teste com p-valor igual à 2.2e-16). O próximo teste realizado foi de Kruskall, o p-valor obtido foi 2.738e-10, o que levou a rejeitar a hipótese nula e afirmar que ao menos dois paradigmas possuem distribuições diferentes. Com o teste par a par Wilcoxon-Mann-Whitney [14], aplica-se em dados com distribuição não normal, validamos que todos possuem distribuição diferentes entre si, como é possível verificar através dos seguintes p-valores:

- 4.929e-06 Funcional e Lógico;
- 2.219e-10 para Imperativo e Funcional;
- 0.0181 para Imperativo e Lógico.

Porém, observando as imagens encontramos uma distribuição atípica. É possível perceber que em **muitos** casos a duração foi

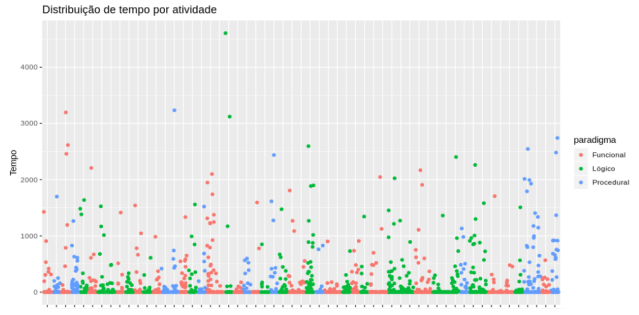

Figura 10: Gráfico de dispersão no qual exibe cada atividade e o tempo em segundos gasto

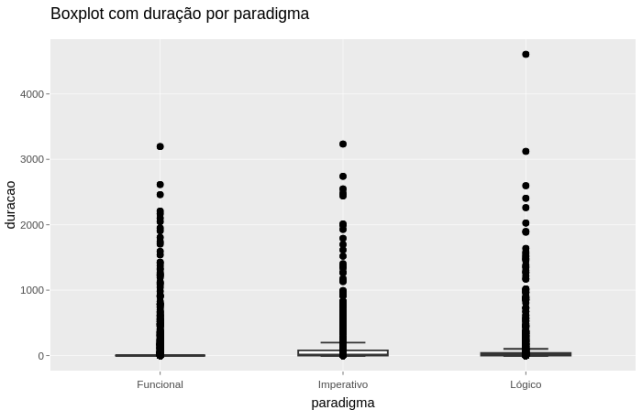

Figura 11: Boxplot do tempo e o paradigma

classificada como 0, o que significa que muitos alunos acertaram na primeira submissão da atividade, pode-se perceber mais claramente na Figura 11. Podemos apontar algumas prováveis causas para esse cenário: os alunos realizaram mais testes locais submetendo apenas quando seguros da resposta; grande domínio no assunto já que várias questões são disponibilizadas para que os alunos possam estudar fora do horário do laboratório; como também podem o auxílio dos colegas os quais já haviam concluído a atividade.

No boxplot, a mediana dos três paradigmas é 0, indicando que mais da metade são casos em que os alunos submeteram corretamente na primeira tentativa. Entretanto, desconsiderando os outliers, os três paradigmas possuem um comportamento semelhante, com o máximo de tempo gasto na faixa dos 3200 segundos, então para ***RQ3*** *podemos dizer que os alunos demoram tempo semelhante para solução dos problemas, independente do paradigma.*

Para os casos no outro extremo, com tempos de resposta muito acima da média, uma possível justificativa é que sejam os casos onde o aluno iniciou a atividade, fez sua primeira submissão sem sucesso e tentou outra atividade, voltando para a atividade e assim submeter corretamente após um longo tempo. Essa situação ocorreu, por exemplo, para uma questão do laboratório de Prolog Básico.

Agrupando o tempo investido na resolução da atividade por laboratório temos a Figura 12, assim como visto por paradigma mais da metade das atividades por alunos (representado pelos pontos no boxplot), são casos que os alunos acertaram na primeira submissão.

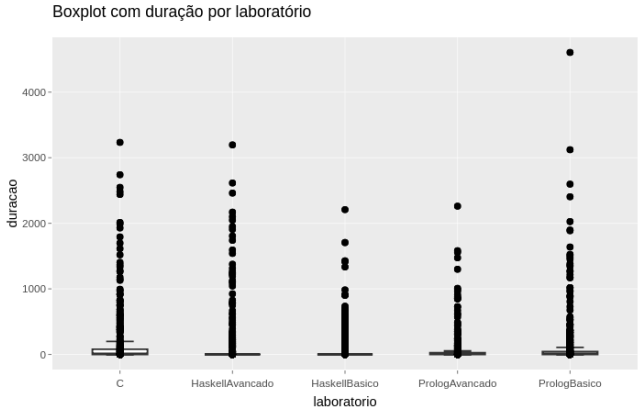

Figura 12: Boxplot do tempo e o laboratório

Contudo, diferente da visualização por paradigma conseguimos ver diferença entre os laboratórios, no caso dos laboratórios Haskell Básico e Prolog Avançado, eles necessitam de menos tempo, enquanto os demais continuam possuindo comportamentos equivalentes entre si.

Buscando uma análise mais clara, foi realizado o teste par a par usando Wilcoxon-Mann-Whitney e encontramos os p-valores, a seguir:

- 3.423e-08 para C e Haskell Básico;
- 1.719e-07 para C e Haskell Avançado;
- 0.0001802 para Haskell Básico e Prolog Básico;
- 0.0003317 para Haskell Avançado e Prolog Básico;
- 0.006974 para Haskell Básico e Prolog Avançado;
- 0.009878 para Haskell Avançado e Prolog Avançado;
- 0.02057 para C e Prolog Avançado;
- 0.07393 para C e Prolog Básico;
- 0.4611 Prolog Básico e Prolog Avançado;
- 0.8931 Haskell Básico e Haskell Avançado

Com esses dados temos que nem todos apresentam distribuições diferentes, se usarmos esses dados como um ranking temos que a maior diferença é entre o laboratório de C e Haskell Básico, isso porque Haskell Básico está entre o que foi necessário menos tempo envolvido e C porque os alunos gastam um tempo maior.

5 CONSIDERAÇÕES FINAIS

Ao longo dos anos vários trabalhos foram realizados visando aperfeiçoar a prática de ensino e melhorar a aprendizagem dos alunos em programação. Este trabalho, buscou-se compreender a dificuldade dos alunos de PLP, com foco nos exercícios de diferentes paradigmas de programação.

Diante disso, as perguntas de pesquisa foram elaboradas com o objetivo de caracterizar o que seria dificuldade, e através das respostas definir o(s) paradigma(s) que representam maior dificuldade para os alunos.

Analisando os dados para responder **RQ1**, foi verificado ao aplicar os testes estatísticos que os mesmos apresentaram distribuições

iguais, sendo assim, não foi possível através dos números de submissões inferir qual paradigma apresenta maior dificuldade. Para **RQ3**, os dados foram analisados e cada paradigma apresentou função de distribuição diferente, entretanto, ao observar o gráfico, percebeu-se que possuem um comportamento similar. Para os três paradigmas a mediana foi 0, e seus valores máximos estão na mesma faixa de valores, com exceção do paradigma Lógico em que um aluno precisou de 4606.202 segundos para resolver.

Buscando prováveis justificativas para que mais da metade das questões os alunos acertassem na primeira submissão, destaca-se que: alguns alunos podem submeter a atividade apenas quando tiverem testado várias vezes localmente e se sentirem confiantes sobre a solução, e que os alunos podem ter tido ajuda dos colegas, os quais já concluíram a atividade.

Quanto à **RQ2**, uma vez validado os dados através dos testes estatísticos, foi possível definir que os paradigmas apresentam distribuições diferentes. Levando em consideração a mediana como valor representativo, o Paradigma Imperativo seguido do Paradigma Lógico, são aqueles com menor coeficiente, indicando que os alunos precisam submeter mais vezes até o sucesso. Ao conversar com os alunos que participaram dessa disciplina, alguns pontos foram levantados que corroboram essa análise.

O fato de que é com o paradigma imperativo o primeiro contato do aluno com a atividade de laboratório e sua estrutura, além de que, mesmo que seja um paradigma visto anteriormente a linguagem pela qual a mesma é praticada que é C ou C++ é bastante diferenciada das linguagens trabalhadas no curso até o período de PLP.

Pode-se inferir também que esse assunto é menos discutido em sala de aula, pois como mencionado anteriormente, os alunos já conheciam e utilizavam esse paradigma em outras disciplinas. Entretanto, pelos resultados, pode-se sugerir maior atenção nesse paradigma. Sobre as questões destacadas como outliers, ambas possuem apenas um exemplo de entrada e saída, e abordam vários assuntos simultâneos. Os alunos destacaram que tiveram dificuldade em conseguir pensar na resolução do problema, independente do paradigma. Nesse ponto, pode-se sugerir que as questões possuam mais exemplos de entrada e saída, ajudando assim no entendimento do problema.

De modo geral, algumas observações foram realizadas, entretanto, não foi possível definir um paradigma específico como aquele que o aluno possui maior dificuldade de acordo com as métricas que foi utilizada, pois as respostas não convergiram em um resultado. Deste modo, em trabalhos futuros, pretende-se analisar outras métricas como as questões, fazer as mesmas análises, mas categorizando por período para uma avaliação mais precisa, e por fim, procurar por um padrão nas atividades que possuem boa aceitação e os outliers dos dados.

REFERÊNCIAS

[1] 2010. TIPOS DE PARADIGMAS DE PROGRAMAÇÃO MAIS COMUNS. (2010). http://tiserviceprovider.blogspot.com/2010/11/tipos-de-paradigmas-de-programacao-mais.html

[2] 2018. Teste t de Student no R. (2018). http://rstudio-pubs-static.s3.amazonaws.com/408638_6679293d4c7a415eaebe00faa3aea0cb.html

[3] 2015. Teste de Shapiro-Wilk para Normalidade. (2015). https://docs.ufpr.br/~aanjos/CE213/ce213/node8.html

[4] Heming; C. 2018. Ferramenta De Apoio Ao Ensino E Aprendizagem De Algoritmos E Programação. (2018). https://www.univates.br/bdu/bitstream/10737/2228/1/2018ClevertonHeming.pdf

[5] Lemos; F Araujo; E Figueiredo; J Guerrero; D Gaudencio; M., Wanderley; L. 2013. Eu Sei o que Voces Fizeram (Agora e) na Aula Passada: o TSTView no Acompanhamento de Exercícios de Programação. (2013). https://www.br-ie.org/pub/index.php/sbie/article/view/2498/2157

[6] Taumaturgo; M. Gomes; D. 2017. Quais as melhores práticas para ensinar programação e como inovar no processo de ensino e aprendizagem? (2017). https://medium.com/cesar-reports/quais-as-melhores-pr%C3%A1ticas-para-ensinar-programa%C3%A7%C3%A3o-e-como-inovar-no-processo-de-ensino-e-42f65694e7e3

[7] Valentim; H. 2009. Um Estudo Sobre O Ensino-aprendizagem De Lógica De Programação. (2009). http://posgrad.fae.ufmg.br/posgrad/viienpec/pdfs/137.pdf

[8] Berssanette; J. 2016. Ensino De Programação De Computadores: Uma Proposta De Abordagem Prática Baseada Em Ausubel. (2016). http://repositorio.utfpr.edu.br/jspui/bitstream/1/2487/1/PG_PPGECT_M_Berssanette%2C%20Jo%C3%A3o%20Henrique_2016.pdf

[9] Coutinho; J. Chagas; F. Moreira; G., Holanda; W. 2018. Desafios na aprendizagem de programação introdutória em cursos de TI da UFERSA, campus Pau dos Ferros: um estudo exploratório. *ECOP 2018, 19 a 21 de Março de 2018, Pau dos Ferros, RN* (2018). https://periodicos.ufersa.edu.br/index.php/ecop

[10] Moreira; L. Paillard; G. 2017. Dificuldades para o ingresso e permanência na ciência e engenharia da computação: um olhar feminino. *Revista Tecnologias na Educação* 19 (2017). http://tecedu.pro.br/wp-content/uploads/2017/07/Art15-vol19-julho2017.pdf

[11] Campos; R. 2010. Metodologia ERM2C: Para melhoria do processo de ensino-aprendizagem de lógica de programação. (2010). http://www.lbd.dcc.ufmg.br/colecoes/wei/2010/0023.pdf

[12] Silva; D. Mattos; G. O.; Moreira J. A. Sales; A., Calado; B. 2014. Dificuldades para o ingresso e permanência na ciência e engenharia da computação: um olhar feminino. *REDOR, UFRPE. Recife, PE* 18 (2014).

[13] 2008. Teste de Kruskal-Wallis. (2008). http://www.leg.ufpr.br/~silvia/CE701/node89.html

[14] 2015. INTRODUÇÃO À ESTATÍSTICA BÁSICA. (2015). http://www.liaaq.ccb.ufsc.br/files/2013/10/Aula-4.pdf

[15] 2016. Linguagens e paradigmas de programação. (2016). https://www.treinaweb.com.br/blog/linguagens-e-paradigmas-de-programacao/